iPosRAID: iSCSI 지원 고성능 스토리지 시스템

류준길° 남영진 김대웅 김도훈 박찬익
포항공과대학교 컴퓨터공학과 시스템소프트웨어연구실
{lancer°, yjnam, woong, hunkim, cipark}@postech.ac.kr

*i*PosRAID: iSCSI-enabled, High-Performance Storage System

Junkil Ryu° Young Jin Nam Daewoong Kim, Dohun Kim, Chanik Park
System Software Laboratory, Department of Computer Science and Engineering, POSTECH

요 약

본 논문은 IP 스토리지 네트워크를 이용한 저장시스템 기술 중에서 활용 범위가 높고 최근에 표준화된 iSCSI (internet SCSI) 프로토콜을 실제 RAID 시스템 내에 구현하고, 그 성능을 기존의 파이버채널(FC) 스토리지 네트워크 기술과 비교 평가한다. 또한, iSCSI 프로토콜을 RAID내에 구현시 고성능을 위해서 고려해야 될 사항들과 그 영향을 분석한다.

1. 서 론

다양한 멀티미디어 응용 및 사회전반적인 분야의 디지털화에 따라, 스토리지 내에 저장해야할 데이터의 양이 기하급수적으로 증가하게 되었다. 기존의 SCSI와 같은 입출력 버스 기반 방식은 이러한 폭증하는 데이터를 수용하기에 성능 및 거리상의 확장성 및 분산되어 존재하는 스토리지의 관리의 복잡성등 많은 문제를 갖게 되었다. 이러한 문제를 극복하고자 스토리지 자원을 일종의 네트웍형태로 연결하는 SAN (Storage Area Network)[2]기술이 출현하게되었다. 이러한 SAN 기술의 대표적인 것이 파이버채널(이하 FC)였다. 그러나 FC는 설치하는 비용이 많이들 뿐만 아니라, 기존에 널리 사용되고 있는 IP 네트워크와는 다른 프로토콜을 사용하는 것이어서 관리하기 어렵다는 문제점이 지적되어 왔다. 따라서,최근에는 FC와 같이 별도 프로토콜을 이용하는 것이 아니라 기반 시설이 잘 갖추어지고 널리 익숙한 *IP* 네트워크를 사용하여 저장시스템을 구축하려고 노력해오고 있다. 이러한 노력 중 표면적으로 나온 것이 FCIP, iFCP, iSCSI 등이다[2]. 그 중에서 iSCSI는 iFCP나 FCIP와는 달리 End-to-End에서 적용할 수 있는 프로토콜이기 때문에 활용 범위가 넓으며, 개발에 따른 기대이익이 크다고 할 수 있다 [2]. 이 논문에서는 이러한 장점을 가지고 있는 iSCSI 프로토콜을 실제로 구현해 봄으로써 디자인 상 고려해야 될 점을 생각해보고, FC상에서 나오는 성능과 비교해봄으로써 *IP* 네트워크 상에서 성능을 최적화 시키기 위해서 고려해볼 사항을 알아본다.

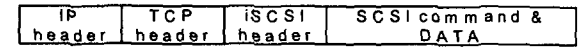
본 논문의 구성은 다음과 같이 구성되어 있다. 2절에서 iSCSI 프로토콜과 그것이 구현되어진 PosRAID에 대해서 간단히 설명하고 구현상 이슈들을 설명한다. 3절에서 FC기반의 PosRAID 성능과 비교해봄으로써 iPosRAID의 문제점과 개선 방안을 생각해 본다. 끝으로, 4절에서 결론과 향후 연구 방향을 기술하도록 한다.

2. iPosRAID[4] 구현

*PosRAID*는 본 연구실에서 개발한 RAID 소프트웨어로서, 실시간 운영체제 모듈, 입출력통신 모듈 그리고 RAID 운영 모듈로 구성되어 있다.([그림2] 참조) 실시간 운영체제 모듈은 상위 모듈들이 원활하게 수행할 수 있는 기반기능을 제공해 주는 것으로써, vxWorks 실시간 운영체제를 기반으로 하고 있다. 입출력 통신 모듈은 호스트 시스템과 의 통신을 담당하는 호스트 입출력 모듈과 내부 디스크와의 통신을 담당하는 디스크 입출력 모듈로 구성되어 있다. 이를 제외한 나머지 모듈들은 실제적으로 각 RAID 레벨을 효과적으로 운영하기 위해서 필요한 모듈로써, 입출력 요구 전처리기 모듈, 버퍼 캐쉬 모듈, 록 관리 모듈, RAID 레벨 종속 모듈, 자원 관리 모듈 등이 있다. PosRAID의 호스트 입출력 모듈에 iSCSI 입출력 모듈을 추가 구현함으로써 IP 네트워크를 통해서 직접 서비스 할 수 있도록 하였다. iSCSI 입출력 모듈이 추가된 PosRAID를 Internet PosRAID란 의미에서 "*iPosRAID*"라고 부르는 것이다.

2.1 iSCSI 프로토콜 소개 ([1],[2])

iSCSI 프로토콜은 SCSI 버스를 통해서 SCSI 명령과 관련 데이터를 보내던 것을 TCP/IP 프로토콜을 이용하는 IP 네트워크를 통해서 보내려고 하는 기술이다. 즉, IP 를 이용함으로 해서 널리 연결되어 있는 IP 네트워크를 활용하고, 상위 계층인 TCP를 사용함으로 해서 신뢰성 있는 연결을 유지할 수 있도록 한다. 그리고 마지막으로 iSCSI 프로토콜 헤더을 이용해서 SCSI 명령이나 데이터가 타겟의 위치를 제대로 찾아가게 한다. ([그림1]참조)



[그림1] iSCSI PDU (Protocol Data Unit)

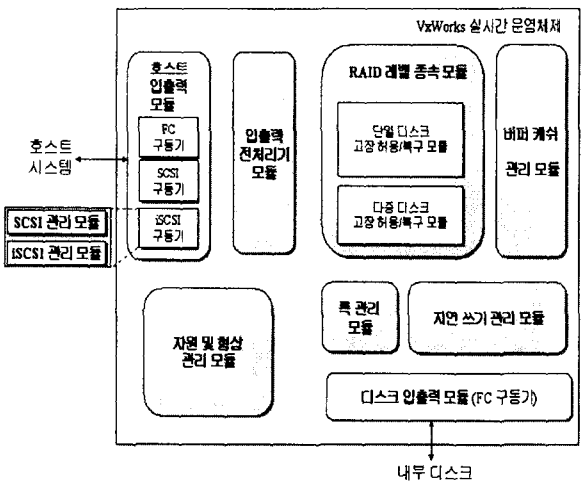
iSCSI 프로토콜은 2000년 7월 드래프트 버전0을 발표하는 것으로 표준화 작업을 시작해서, 2003년 2월에 표준화 작업이 완료되었다. iPosRAID는 iSCSI 드래프트 버전 12 (2002년 9월 현재)를 기준으로 하여 개발되어졌다.

2.2 구현상 특징

네트워크를 통해서 데이터를 받을 때, 일어나는 메모리 복사는 고속의 데이터 전송 시스템에서 CPU자원을 소비하는 중요한 원인이다. 현재 본 연구실의 *iPosRAID* 구현환경은 기가비트 이더넷 환경을 기반으로 하고 있기 때문에 데이터를 주고받을 때, 생기는 메모리 복사에 따른 CPU자원 소모가 크다. 따라서 구현에 있어서는 vxWorks에서 제공하는 제로 카피(zero copy)를 지원하는 zBuf를 사용하여 네트워크 프로토콜을 처리할 때 발생하는 CPU 자원의 소모를 최소화 시키도록 하였다.

[그림2]에서 보는 것처럼 iSCSI는 PosRAID의 호스트 입출력 모듈로서 개발되어졌다. *iSCSI* 입출력 모듈은 세션을 생성, *iSCSI* PDU 헤더를 분석하는 iSCSI 관리 모듈과 RAID 엔진으로 SCSI 명령을 내리는 SCSI 관리 모듈로 구성되어진다. iSCSI 호스트 입출

럭 모듈은 iSCSI PDU를 받아서 iSCSI 프로토콜 헤더를 분석하고 그것에 맞게 SCSI 명령이나 데이터를 분리해낸다. 그리고 그것을 PosRAID에 입출력 전처리기에 보내게 되는데, 기능상 iSCSI 프로토콜 헤더를 분석 처리하는 것과 그에 해당하는 SCSI 명령이나 데이터를 PosRAID에 보내는 것은 구분이 되고, 전자가 네트워크(TCP/IP)에 관련된 부분이 많은 반면, 후자는 PosRAID에 관련된 부분이 많기 때문에 iSCSI 관리 모듈과 SCSI 관리 모듈로 분리해서 구현하였다.



[그림2] iPosRAID Layout

성능을 고려할 때에도, 하나의 부분으로 구현하였을 때는 iSCSI PDU 헤더를 분석하고 그에 해당하는 SCSI 명령을 PosRAID 엔진에 보내고 그에 대한 반응(Response)을 호스트에게 보내는 것이 일괄적으로 일어나기 때문에 여러 개의 호스트가 연결될 때는 성능 저하를 가져 올 수 있다. 실제 iSCSI PDU을 분석하는 부분의 오버헤드가 가장 크기 때문에 이것을 분리하는 것이 여러 개 호스트의 연결을 지원할 때 반응시간을 줄일 수 있는 방법이다.

iSCSI 관리 모듈은 호스트로부터 연결 요청을 받아들이는 ServerTask 쓰레드와 그 연결을 관리하는 Rx/Tx 쓰레드로 구성되고, SCSI 관리 모듈에는 PosRAID 엔진으로 명령을 보내는 것을 관리하는 쓰레드로 구성된다.

2.3.1 iSCSI 관리 모듈

현재 구현에서는 연결 요청이 들어오면 연결을 관리할 두 개의 쓰레드 (Rx/Tx Thread)를 생성하고, 하나의 TCP 연결로 하나의 세션을 생성한다. 그러나 이것은 성능상의 문제뿐만 아니라 세션 복구를 위해서도 수정을 해야 할 부분이다. 즉 하나의 세션이 여러 개의 연결로 구성될 수 있도록 별도의 세션 쓰레드를 두어 여러 개의 Rx/Tx Thread들을 관리할 수 있게 만들어야 될 뿐만 아니라 여러 연결을 통해서 들어오는 iSCSI PDU (Protocol Data Unit)의 순서를 관리해야 하고 연결이 끊어지는 것에 대비해야한다.

2.3.2 SCSI 관리 모듈

SCSI 관리 모듈에는 PosRAID 엔진과 iSCSI 관리 모듈 사이에서 SCSI 명령과 데이터를 다루는 쓰레드 (scsiTargetThread)가 존재하는데, iSCSI 관리 모듈에 있는 Rx/Tx 쓰레드와는 달리 연결이 생길 때마다 생성되는 것이 아니라 연결되는 호스트의 수에 관계없이 하나만 존재 하게 된다. 여러 호스트가 연결될 때, SCSI 관리 모듈에서 병목현상이 일어날 가능성이 있는데, 실제 테스트 해 본 결과 iSCSI 관리 모듈에서 iSCSI PDU을 분석해서 SCSI 관리 모듈로 전달하는데 걸리는 오버헤드보다는 상당히 작다는 것을 알 수 있었다. 따라서 호스트와 연결이 될 때마다 SCSI 관리 모듈의 관리 쓰레드를 생성하는 것은 오히려 오버헤드를 증가 시키는 것이다. 그렇다고 하나를 가지고 유지하는 것은 많은 호스트와 연결이 될 때 병목현상이 일어날 것으로 예측이 되는 바, 연결되는 호스트의 수가 임의의 수에 도달하면 쓰레드를 생성시켜주는 것도 좋은 방법이 될 것이다.

3. 성능 평가

성능 평가에서는 기가비트 이더넷 상에서 iSCSI 프로토콜이 구현된 iPosRAID와 FC 기반PosRAID 성능을 측정해보고, 그 결과에 대해서 분석을 하겠다. 성능 평가를 위한 실험 환경은 다음과 같다.

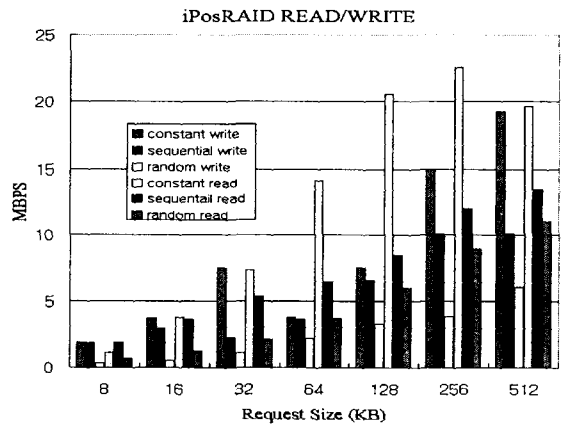
	Linux Hosts	Storage System
CPU	Pentium III 1GHz	Pentium III 1GHz
Memory	512MB	512MB
NIC	1Gbps (Intel pro1000T)	1Gbps (Intel pro1000T)
OS	Linux 2.4.13	vxWorks 5.4

<iPosRAID 실험 설정>

iPosRAID를 위한 실험을 위해, 시스템들은 기가비트 스위치로 연결되었으며, 리눅스에 있는 iSCSI initiator driver는 "University of New Hamshire"의 InterOperability Lab.에서 구현한 것[1]을 이용하였다.[3] 스토리지 시스템의 PosRAID 설정은 [RAID level: 0, stripe unit size: 32KB, logical unit size : 10GB, #of disks: 4] 이다.

	Linux Hosts	Storage System
CPU	Pentium III 1GHz	Pentium III 1GHz
Memory	512MB	512MB
FC adapter	Qlogic 2100	Qlogic 2200
OS	Linux 2.4.13	vxWorks 5.4

<PosRAID 실험 설정>



[그림3] iPosRAID READ/WRITE 성능

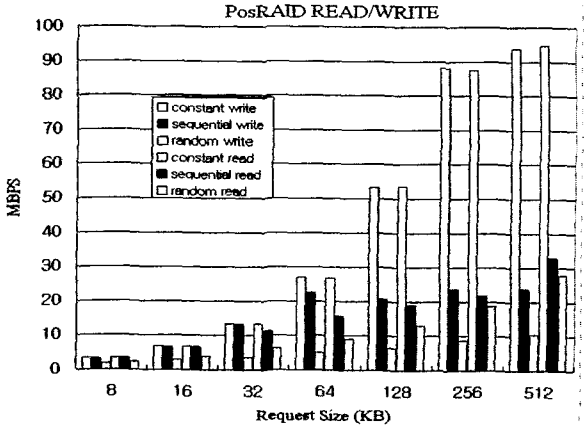
PosRAID의 실험은 스토리지 시스템을 리눅스 호스트에 FC 어댑터와 FC 케이블를 사용하여 직접 연결하였다. 스토리지 시스템의 PosRAID 설정은 iPosRAID 실험할 때와 동일하다. 성능 측정 도구로서는 IOgen[2]을 사용하였다.

[그림3]은 iPosRAID의 READ/WRITE의 성능을 측정한 값이다.

1)iSCSI 프로토콜의 드래프트 버전 테스트를 위해서 구현 된 것이기 때문에 iPosRAID와 호환을 위해서 일부 변경을 하였다.

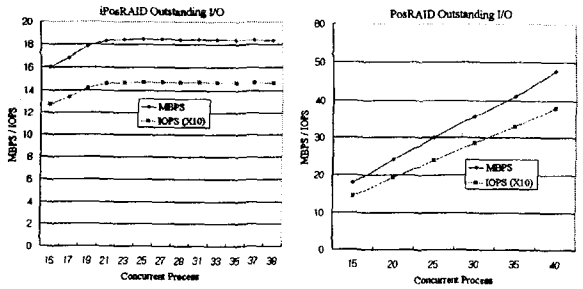
2)본 연구실에서 자체 개발한 Linux SCSI Generic Interface를 이용한

표는 constant, sequential, random순으로 표시되어 있다. cash hit이 일어나는 constant read의 경우 request size가 256KB일때, 23MB/s, Constant write의 경우 request size가 512KB의 경우 19MB/s 정도 나오는 것을 알 수 있다. [그림4]는 FC상의 PosRAID 성능을 측정한 것이다. cash hit이 일어나는 constant read / write 의 경우, request 크기가 512KB인 경우, 각각 94.62MB/s, 93.69MB/s이다. 그러나 sequential이나 random의 경우에는 iPosRAID와 PosRAID의 차이는 줄어든다. 이것은 스토리지 시스템에 있는 디스크의 성능에 많이 의존하기 때문이다.



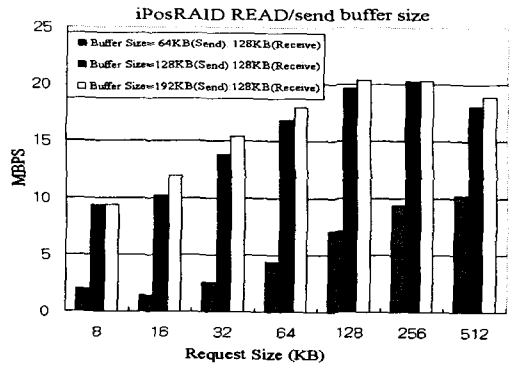
[그림4] PosRAID READ/WRITE 성능

그럼 이러한 성능의 차이가 일어나는 원인에 대해서 알아보자. [그림5,6][3]은 RAID시스템에 I/O requests를 내리는 프로세스의 수를 증가시킴에 따라 throughput (MB/s)와 IO의 수의 증가를 표시한 것이다. 이것을 통해서 iPosRAID의 세션을 구성하는 1개의 TCP connection이 bottle neck인지를 판단하려고 한다. PosRAID의 경우 I/O request를 발생시키는 프로세스의 수가 증가함에 따라 RAID시스템으로 내려가는 초당 I/O request의 수와 throughput이 증가함에 비해, iPosRAID의 경우에는 그렇지 못함을 알 수 있다.



[그림5] iPosRAID에서 I/O 프로세스 수에 따른 MB/s와 IO의 수 [그림6] PosRAID에서 I/O 프로세스 수에 따른 MB/s와 IO의 수

이것은 iPosRAID의 세션을 구성하는 TCP 연결이 하나로 이루어져 있어서 많은 IO request를 RAID 시스템에 보내지 못하기 때문이다. 이것을 간접적으로 확인하기 위해서 두개의 호스트를 연결하여 I/O request를 내렸을 때, constant request의 경우 한 개를 연결했을 때와 거의 같은 성능이 나오는 것을 확인 할 수 있었다. Sequential과 random requests의 경우 감소는 했지만 전체적으로는 증가함을 알 수 있었다. 따라서 세션을 이루는 TCP connection의 수를 증가시킬 수 있도록 수정한다면 더욱 좋은 성능을 낼 수 있을 것이다.



[그림7] TCP socket 버퍼 크기 변화에 따른 iPosRAID 성능

[그림7]은 iPOSRAID의 TCP socket의 send 버퍼 크기를 변화시키는 것이 성능에 어느 정도 영향을 미치는 가를 알아보기 위한 것이다. [그림7]의 경우 버퍼 크기가 일정 수준이상 크면 그것의 영향은 그다지 크지 않음을 알 수 있다. 그러나 버퍼 크기를 64KB에서 128KB로 커질 때 성능 차이는 크다는 것을 알 수 있다. 이것은 RAID의 stripe unit size가 32KB (32 X 4 =128)이기 때문이다.

4. 결론 및 향후 계획

최근에 표준화된 iSCSI를 실제 시스템에 구현해본 결과, 수정해야 될 부분이 있지만, 기대해도 될 만한 성능을 보인다는 것을 알 수 있다. 따라서 향후 계획으로는 iSCSI 입출력 모듈이 최적으로 작동되게, 세션당 여러 개의 TCP 연결을 생성할 수 있도록 하는 기능, 여러 연결로부터 들어오는 데이터의 흐름을 관리할 수 있는 기능 그리고 TCP 연결에 에러 발생시 이것을 복구해 줄 수 있는 기능을 구현하기위해서 세션 관리 모듈을 추가로 작성해야 될 것이다. 또한 데이터의 보안을 위해서 iSCSI모듈에 IPSec를 지원하도록 하는 것이다.

5. 참고 문헌

[1] J. Satran, et al., *"iSCSI draft standard"*, http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-12.txt , 2002.

[2] Tom Clark, *"IP SANs : A Guide to iSCSI, iFCP, and FCIP protocols for Storage Area Network"*, Addison-Wesley, 2002.

[3] InterOperability Lab.,UNH, "iSCSI Consortium" http://www.iol.unh.edu/consortiums/iscsi/, 2002.

[4] Y.J.Nam and et. al., *"Enhancing Write I/O Performance of Disc Array RM2 Tolerating Double Disk Failure"*, Proceedings of ICPP, Aug. 2002.

스토리지 성능 벤치마크 툴이다.

3) [그림5,6]의 Request Size는 128KB이고, I/O request는 constant read이다.